UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO

Formulações Exata e Heurística para o Problema de Programação de Horários da Universidade Federal de Sergipe

Trabalho de Conclusão de Curso

Dimitri Carvalho Menezes

São Cristóvão – Sergipe

2017

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Dimitri Carvalho Menezes

Formulações Exata e Heurística para o Problema de Programação de Horários da Universidade Federal de Sergipe

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Renê Pereira de Gusmão

São Cristóvão – Sergipe

2017

Agradecimentos

A deus, por ter me dado força para superar as dificuldades e por me manter mentalmente saudável até o fim do curso.

Aos meus pais, Sônia e Adriano, minhã irmã Angelis, Raimunda e a todos os meus familiares, por todo esforço que fizeram por mim.

A Universidade Federal de Sergipe e ao Departamento de Computação, pela oportunidade de realizar o curso de Ciência da Computação.

Ao meu orientador Renê, pelo empenho e paciência na elaboração deste trabalho.

"Ever tried. Ever failed. No matter. Try Again. Fail again. Fail better."

[1] Samuel Beckett

[1] Tradução: Tenta. Fracassa. Não importa. Tenta outra vez. Fracassa de novo. Fracassa melhor.

Resumo

O problema de programação de horários baseado em currículos é um problema de otimização combinatória e considerado como um problema NP difícil. A grande dificuldade desse problema é alocar aulas para professores, salas e horários sem que haja conflitos de tempo e espaço. Basear-se em currículos ou matriz curricular significa que os horários das aulas de um currículo não devem estar em conflitos. Esse problema é comum em universidades e uma formulação geral do problema acaba não sendo útil para todas as universidades, pois as regras e restrições institucionais mudam de uma universidade para outra. A construção de um modelo matemático pode servir como base para novos estudos ou até mesmo a construção de uma ferramenta que facilite a organização dos horários dos professores de uma universidade. Logo este trabalho teve como objetivo construir um modelo de programação linear inteira para a Universidade Federal de Sergipe com a ideia de maximizar a preferência dos professores em lecionar aulas em horários específicos. E para verificar o quão difícil é solucionar o problema complementado com as restrições específica da universidade, foi utilizado o método exato de branch and bound e a metaheurística Iterated Local Search. Os resultados mostram que o método exato consegue obter soluções após utilizar muito tempo e recursos computacionais. Porém, não foi possível provar a otimalidade das soluções dentro do limite de tempo determinado. Enquanto que o método heurístico consegue obter soluções próximas ou melhores do que o método exato, utilizando menos tempo e recursos computacionais.

Palavras-chave: Otimização combinatória, programação de horários, matriz curricular, restrições, branch and bound, método heurístico , programação linear inteira

Abstract

Curriculum based timetabling problem is a combinatorial optimization problem and considered as a NP-Hard problem. The biggest difficulty of this problem is to allocate classes for teachers, classrooms and schedules without conflicts of time and space. Relying on curriculum or curriculum matrix means that curriculum schedules should not be in conflict. This is a usual problem in universities and a general formulation is not a good deal to all universities, as institutional rules and restrictions change from one university to another. The construction of a mathematical model can be the basis for new studies or even the construction of a tool that simplifies the organization of teachers schedules of a university.Therefore, this study aimed to build a specific model for the Federal University of Sergipe with the idea of maximizing teachers' preferences to teach classes at specific time. In addition, to verify how difficult it is to solve this problem increased with the specific restrictions of the university, it was used branch and bound exact method and Iterated Local search heuristic method. The results show that the exact method can obtain solutions after using a lot of time and computational resources. However, it was not possible to prove the optimality of the solutions within the given time limit. While the heuristic method can obtain solutions close to or better than the exact method, using less time and computational resources.

Keywords: Combinatorial optimization, timetabling problem, curricular matrix, constraints, branch and bound, heuristic method, integer linear programming

Lista de ilustrações

Figura 1 – Vizinhança de soluções 38
Figura 2 – Soluções encontradas do CN2 em relação ao tempo 40
Figura 3 – ILS com swapMove para o cenário 2 41
Figura 4 – ILS com singleMove para o cenário 2 42
Figura 5 – ILS com doubleMove para o cenário 2 42
Figura 6 – Resultados do CN3 com os métodos de perturbação 43

Lista de tabelas

Tabela 1 – Comparação entre os principais formulações de restrições definidos para o CBCT 20
Tabela 2 – Relação Dia/Horários letivos da Universidade Federal de Sergipe 28
Tabela 3 – Comparação das restrições usadas no modelo UFS com os modelos bases de Gapero 2007 e Bonutti 2012 32
Tabela 4 – Resultados das execuções dos cenários do método exato 40
Tabela 5 – Comparação dos resultados das execuções dos cenários com diferentes perturbações 43

Lista de Algoritmos

1 Pseudocódigo ILS 23
2 Gerar solução inicial 34
3 Busca Local 35
4 Perturbação: singleMove 36
5 Perturbação: doubleMove 36
6 Perturbação: swapMove 37

Lista de abreviaturas e siglas

CBCT	Curriculum-Based Course Timetabling
DAA	Departamento de Administração Acadêmica
ITC07	Internacional Timetabling Competitions 2007
ILS	Iterated Local Search
GRASP	Greedy Randomized Adaptive Search Procedure
OPL	Optimization Programming Language
PPAS	Problema de Alocação de Aulas a Salas
PPH	Problema de Programação de Horários
PPHE	Problema de Programação de Horários Educacionais
PPHU	Problema de Programação de Horários Universitários
PPTC	Problema de Programação de Horários Professores x Turmas com restrições de Compacidade
SIGAA	Sistema Integrado de Gestão de Atividades Acadêmicas
UFS	Universidade Federal de Sergipe

Sumário

1 Introdução 11
1.1 Objetivos 14
1.2 Justificativa 14
1.3 Estrutura do Documento 15

2 Fundamentação Teórica 16
2.1 Definição formal de um problema de programação linear 16
2.2 Programação Inteira 17
2.3 O Problema de Programação de Horários baseado em Currículo 18
2.4 IBM iLOG CPLEX Studio e a Optimization Programming Language 21
2.5 O algoritmo branch and bound 22
2.6 A metaheurística Iterated Local Search 23
2.6.1 Gerando a solução inicial 23
2.6.2 O método de busca local 24
2.6.3 O método de perturbação 24
2.6.4 Os critérios de aceitação 24
2.7 Trabalhos Relacionados 25

3 Metodologia 27
3.1 Formulação do problema 27
3.2 Entrada de dados 32
3.3 O algoritmo ILS construído 33
3.3.1 Construção da solução inicial 33
3.3.2 Método de busca local 34
3.3.3 Perturbação: SingleMove 35
3.3.4 Perturbação: DoubleMove 36
3.3.5 Perturbação: SwapMove 36
3.3.6 Vizinhança de soluções 37

4 Resultados 39
4.1 Execução do método exato 39
4.2 Execução do método heurístico 41

5 Conclusão 45

Referências 47

1

Introdução

Periodicamente, nas universidades, é necessário a construção de quadros de horários composto por turmas e professores para diversas disciplinas. Criar esses quadros é uma tarefa complicada e ocorrem situações em que até mesmo os coordenadores de cursos gastam dias fazendo-os à mão. Os planos de semestres anteriores das universidades servem como base, porém nem sempre é possível reaproveitar devido ao fato de que as preferências por disciplinas e horários de cada professor mudam ao longo dos semestres conforme às necessidades dos departamentos e questões pessoais.

Em adição, contratações e recisões de contratos dos professores, adição ou remoção de componentes curriculares dos curso, inativação de uma sala de aula ocorrem com frequência em um departamento. Um outro fator que dificulta a tarefa de planejamento de quadro de horários é basear-se em outra instituição, pois a estrutura de uma instituição é distinta das outras devido a diferenças na carga horária letiva, regras de ofertas de disciplinas e outros aspectos próprios de cada instituição, de modo que não exista um padrão que possa ser aplicada a todos os casos (BARBOSA, 2012).

O problema descrito é um problema matemático de difícil resolução chamado de Problema de Programação de Horários PPH, do inglês *timetabling problem*, tendo os estudos iniciados na década de 1970. É classificado por Even, Itai e Shamir (1975) como um problema NP-completo, pois os algoritmos conhecidos resolvem o problema em tempo polinomial não determinístico. Wren (1996) definiu o PPH como:

> *"A alocação, sujeita a restrições, de recursos a objetos colocados no espaço e no tempo, de modo a satisfazer, tanto quanto possível, um conjunto de objetivos desejáveis."*

O PPH envolve qualquer atividade de agendamento e construção de quadro de horários.

Soluções gerais podem ser aplicadas em áreas de indústrias, transportes, hospitais, instituições de ensino, e eventos esportivos (WILLEMEN, 2002). Especificamente, para instituições de ensino, o problema é denominado de Problema de Programação de Horários Escolares (PPHE). Este problema envolve esquematizar os horários dos professores em turmas do ensino fundamental e ensino médio. Apesar de ser um problema complexo, há muita relevância em estudar sobre o assunto pois a construção de quadro de horários de forma eficiente melhoraria tanto as atividades dos professores para com as turmas quanto os recursos como horários, salas e laboratórios fornecidos pelas instituições.Existem outras classificações para o PPHE segundo Santos e Souza (2007):

- Problema de Programação de Horários de Exames em Universidades (PPHEU), no qual o objetivo é agendar exames para as turmas sem conflitos de tempo e espaço.
- Problema de Programação de Horários de Cursos em Universidades (PPHU), consiste em determinar o horário e local onde os cursos de diversas turmas serão realizados.
- Problema de Alocação de Aulas a Salas (PAAS), trata-se da alocação de aulas em salas com horários definidos. Pode ser tratado junto com a programação de horários ou após a matrícula.
- Problema de Programação de Horários Professores x Turmas com restrições de Compacidade (PPTC), relacionado a satisfação dos professores com o quadro de horários, especificamente a compacidade de horários, no qual a minimização de janelas de horários e carga horária máxima diária de atividade acadêmica são os fatores mais importantes.

A realização da *Internacional Timetabling Competitions 2007* ITC07 teve como objeto de estudos o PPHE e outras variações, e os modelos matemáticos propostos pelos pesquisadores se tornaram importantes referências. Isso se confirma com a grande quantidade de publicações feitas utilizando estes modelos (ver seção 2.7). A competição acrescentou aos estudos do PPHE duas importantes categorias:

- Problema de Programação de Horários baseado em Currículo do inglês *Curriculum based Course Timetabling* - CBCT estudado por Gaspero, McCollum e Schaerf (2007) no qual os conflitos são determinados de acordo com os currículos das disciplinas. Com isso, as disciplinas que pertencem ao mesmo currículo não devem conflitar horários. Em adição, a grade de horário é definida antes que os estudantes se registrem nos cursos.
- Problema de Programação de Horários Pós Matricula, estudado por Lewis, Paechter e McCollum (2007) onde os conflitos entre cursos são resolvidos após matrícula dos estudantes nos determinados cursos.

Para a proposta descrita nesse documento, o problema será restrito ao Problema de Programação baseado em Currículo. O grande desafio do CBCT é programar uma série de encontros, geralmente no período letivo semanal, entre professores e alunos, obedecendo regras institucionais (ex: duração de aulas), limites pedagógicos (ex: capacidade de salas e laboratórios para as turmas), determinação das disciplinas de um currículo e satisfação pessoal do professor (ex: preferências por horários e limite de carga horária).

O estudo desse problema requer uma formulação matemática. A formulação matemática é composta por elementos como restrições, variáveis de decisão e uma função objetivo. As restrições impõem condições e delimitam os valores a serem obtidos. As variáveis de decisão são responsáveis por assumir os dados validados pelas restrições e/ou substituir os dados em caso de melhorias na solução. Já a função objetivo avalia a solução obtida e é uma expressão a ser maximizada ou minimizada, submetidas a várias restrições. Portanto, a solução ótima gerada deve possuir o maior ou o menor valor possível relativo a função objetivo.

O CBCT pode ser modelado como um problema de programação linear, segundo Gaspero, McCollum e Schaerf (2007) e é um problema NP-difícil, como afirma Barbosa (2012). Um modelo de programação linear é um modelo matemático no qual a função objetivo e as restrições são definidas como equações ou inequações lineares (GOLDBARG; LUNA, 2005). A formulação matemática do problema requer uma definição de domínios das restrições, variáveis e função objetivo do problema, podendo assumir valores reais, inteiros, booleanos ou mistos. Na programação linear inteira, as variáveis de decisão assumem números inteiros. Utilizam-se as variáveis mistas quando nem todas variáveis podem ser definidas como inteiros.

Definida uma representação matemática do problema, implementa-se essa representação em forma de algoritmo, utilizando métodos exatos ou métodos aproximados (heurísticos). Os métodos exatos consistem em enumerar todas as possíveis soluções e escolher a solução ótima. Há diversas técnicas de soluções exatas. As principais são: os algoritmos simplex, algoritmos de grafos, programação dinâmica, programação com restrições e *branch and bound*. Em certos casos, a utilização deste método torna-se inviável mesmo para alguns problemas pequenos, uma vez que é necessário muito tempo computacional para enumerar todas as soluções possíveis (JARDIM; SEMAAN; PENNA, 2016).

Utilizar o método exato, a princípio, é importante para questões experimentais pois serve para observar em quais situações e cenários o método se torna inviável, antes de utilizar métodos heurísticos que buscam soluções boas suficientes em tempo reduzido (JARDIM; SEMAAN; PENNA, 2016). Alguma das principais heurísticas e meta-heurísticas são: Algoritmos genéticos, busca tabu, busca local, *GRASP - Greedy Randomized Adaptive Search Procedures*, redes neurais e *Simulated Annealing*.

1.1 Objetivos

Este trabalho tem como objetivo geral desenvolver uma formulação matemática utilizando programação linear inteira do Problema de Programação de Horários Baseado em Currículo da Universidade Federal de Sergipe. Os objetivos específicos são:

- Executar a formulação matemática utilizando um método exato;
- Construir uma meta-heurística baseada na formulação;
- Comparar o desempenho do método exato baseado em programação linear inteira com o desempenho da meta-heurística, em termos de qualidade das soluções geradas e tempo de execução.

1.2 Justificativa

O que impulsionou a realização desse trabalho foi entender que os problemas de programação de horários, no geral, são de difícil resolução e de grande interesse da comunidade de Pesquisa Operacional, pois os resultados obtidos melhoram o processo de construção de quadro de horários das universidades. Apesar dos resultados serem soluções comuns para as universidades, uma formulação matemática geral acaba não sendo útil pois restrições e a função objetivo da formulação mudam de uma universidade para outra.

Através de entrevistas com professores, coordenadores e diretor do Departamento de Assuntos Acadêmicos - DAA e um dos responsáveis pelo processamento de matrícula da Universidade Federal de Sergipe, verificou-se o quão complexo é construção de grades de horários da universidade. Atualmente na UFS o processo de construção das matrizes curriculares ocorre da seguinte forma: os coordenadores dos cursos listam as disciplinas que atendam a necessidade dos cursos e repassem aos professores. Estes devem listar todas as disciplinas e os horários de sua preferência e repassam as informações para os coordenadores dos cursos. Os coordenadores têm acesso ao módulo do SIGAA, que é responsável por processar as ofertas das turmas, e tenta criar uma nova turma lecionada pelo professor em determinado horário. Caso já exista turmas da mesma matriz curricular no mesmo horário, o sistema reporta uma mensagem de erro do conflito ocorrido. Os coordenadores se reunem com os professores e estes negociam e resolvem a situação.

Esse processo de definir a matriz curricular, em média, dura mais de duas semanas até todas as alocações das turmas para professores e horários serem concluídas. Já a distribuição das turmas nas salas é feita pelo DAA - Deparamento de Assuntos Acadêmicos. O professor Dr. Antônio Edilson do Nascimento é responsável por coordenar as pessoas responsáveis por selecionar manualmente todas as turmas com quantidade de alunos compatível com as salas. Os coordenadores podem adicionar turmas extras em casos de necessidades. Porém, esta etapa

geralmente ocorre sem a verificação de conflitos nos horários. Além disso, o coordenador deve procurar um professor que esteja disponível e apto a lecionar a turma.

A formulação matemática criada serve como um primeiro estudo do problema relacionado a UFS. O modelo matemático de programação linear inteira utilizado como base é o de Gaspero, McCollum e Schaerf (2007), porém modificado com adições de novas restrições, não utilização de algumas restrições e uma nova função objetivo. Utilizar o método exato antes do método heurístico visa obter parâmetros para comparações de eficiência dos dois métodos. Além disso, é necessário uma análise do método exato primeiro para verificar em quais cenários ele é inviável e assim utilizar o método heurístico para solucionar o problema de forma mais rápida.

O modelo matemático de programação linear que será criado pode ser utilizado em um trabalho futuro na construção de uma ferramenta de suporte para os professores, e/ou uma ferramenta web para a própria universidade. Definir ofertas de turmas e horários de forma automatizada facilitaria aos professores gerenciarem suas atividades e as instituições organizarem seus recursos tais como o corpo docente de seus departamentos.

1.3 Estrutura do Documento

Para organizar e melhorar o entendimento, este documento está estruturado em capítulos e seções. O capítulo dois levanta os principais conceitos como programação linear inteira, método exato, meta-heurística e a definição do CBCT, alem dos trabalhos relacionados que dão base para a proposta levantada. O capitulo três detalha o modelo de programação linear inteira construído. O capítulo quatro mostra as execuções e os resultados obtidos. Por fim, o capítulo cinco traz a conclusão e expectativa para trabalhos futuros.

2

Fundamentação Teórica

Essa seção apresenta os conceitos básicos que dão base para este trabalho. A primeira seção retrata como um problema de otimização combinatória pode ser representado matematicamente através de programação linear. Após isso, as principais características do problema são apresentadas. Em seguida, os detalhes da ferramenta *IBM ilOG CPLEX STUDIO* e do método exato *branch and bound*. Por fim, a meta-heurística ILS é apresentada.

2.1 Definição formal de um problema de programação linear

Um problema de otimização geralmente envolve alocar recursos em atividades. Os recursos podem ser dinheiro, máquinas, pessoas, veículos e etc. Exemplos de atividades são gerenciamento financeiro de fases de projetos, agendamentos de funcionários em atividades e definição de rotas para veículos. Goldbarg e Luna (2005) e Frederick e Hillier (1986) definem de maneira semelhante uma fórmula geral de programação linear para os problemas de otimização. Dado uma matriz A , um conjunto de dados representado pelo vetores b e c como dados de entrada. A função objetivo é definida como:

Otimizar:

$$z = \sum_{j=1}^{n} c_j x_j \tag{2.1}$$

sujeito a:

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, i = 1, 2, ...p \tag{2.2}$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, i = p+1, p+2, ..m, \tag{2.3}$$

$$x_j \geq 0, j = 1, 2, ..., q \tag{2.4}$$

$$x_j \in \mathbb{R}, j = q+1, q+2, ..., n \tag{2.5}$$

No qual,

z = é o valor máximo ou mínimo a ser obtido;

M = $\{1,2,...,i,...,m\}$ representa os índices dos recursos;

N = $\{1,2,...,j,...,n\}$ representa os índices das atividades;

A = $\{a_{ij}\}$ representa a matriz de recursos i associado por cada atividade j;

$x = x_j$, j $\in$ N variável de decisão de alocação de atividades j

$c = c_j$, j $\in$ N fator peso.

$b = b_i$, i $\in$ M , valor de entrada relacionado aos recursos.

A função objetivo é o maior ou o menor valor de critério z. As restrições (2.2) e (2.3) são chamadas de restrições funcionais pois influenciam diretamente na alocação ou não dos recursos. (2.4) e (2.5) são restrições de domínio. A restrição (2.4) apenas restringe o domínio dos valores do conjunto x aos não negativos. Enquanto que (2.5) restringe os valores do conjunto x a pertencerem aos números reais.

Goldbarg e Luna (2005) apresenta uma abstração mais simples e de forma equivalente. A formulação pode ser expressa na seguinte forma canônica:

Otimizar $z = cx$

sujeito a:

$a_{ij}x_j \leq b_i$ ou $a_{ij}x_j \geq b_i$

$x_j \geq 0$

2.2 Programação Inteira

Programação inteira é um problema da programação linear no qual as restrições de domínio das variáveis são de domínio inteiro. Pode ser considerada programação pura inteira se todas as variáveis de decisão são definidas como inteiro e programação inteira mista se somente algumas dessas variáveis são inteiras.

Comparada com programação linear, o problema torna-se mais complexo pois programação inteira é o problema de programação linear incrementado com as restrições de domínio inteiro, sem relaxação linear. Segundo (WINSTON; GOLDBERG, 2004), as soluções obtidas em programação linear omitindo as restrições de domínio inteiro ou 0-1 das variáveis é chamado de relaxação linear do problema de programação linear inteira. Isso significa que as soluções viáveis da programação inteira estão contidas também na programação linear com relaxação

correspondente. Portanto, o valor de uma solução ótima obtida da programação linear relaxada é igual ou superior ao valor da solução ótima obtida na programação inteira.

Do problema descrito na seção anterior, adapta-se o o problema de programação linear para programação inteira adicionando a restrição de domínio inteiro para a variável x_j:

Otimizar $z = cx$

sujeito a:

$a_{ij}x_j \leq b_i$ ou $a_{ij}x_j \geq b_i$

$x_j \geq 0$, x_j é inteiro

2.3 O Problema de Programação de Horários baseado em Currículo

O problema de programação de cursos baseado em currículo é uma das variações do PPHE, e segundo Barbosa (2012), é um problema NP-difícil. O problema consiste em alocar professores em turmas de disciplinas, dado um número de salas e horários, no qual os conflitos entre os cursos são resolvidos de acordo com os currículos feitos pela universidade. Os principais conjuntos de dados envolvidos de acordo com Barbosa (2012) e Gaspero, McCollum e Schaerf (2007) são:

- **Dias, Horários e Períodos:** dado um número de dias letivos, cada dia é dividido em um certo número de períodos, sendo um horário correspondente ao par dia/período; O numero total de períodos é calculado pelo produto dos dias com os horários.
- **Currículo:** é um grupo de disciplinas no qual muitos estudantes têm em comum. É mais conhecido como uma turma de semestre.(Ex: As disciplinas de uma turma de estudantes calouros). Portanto as disciplinas de um currículo não podem estar em conflitos de horários.
- **Disciplinas e Professores:** cada disciplina apresenta um professor alocado para lecionar (previamente alocado pela instituição); o número de aulas; o número de alunos; o mínimo de dias exigidos na alocação; e, também, um conjunto de horários em que a mesma não está disponível para alocação;
- **Salas:** cada sala apresenta sua respectiva capacidade e, em alguns problemas, sua localização.

Um modelo matemático para o problema é composto por uma função objetivo e restrições. Existem dois tipos de restrições de acordo com Even, Itai e Shamir (1975), Cooper e Kingston (1996) e Santos e Souza (2007):

Restrições fortes: são restrições que validam soluções ou invalidam uma solução em caso de não serem atendidas. Geralmente são restrições de conflitos físicos.

Restrições fracas: são restrições que não invalidam uma solução em caso de não serem atendidas, mas servem como medida de qualidade. Quanto mais restrições fracas atendidas, melhor a solução buscada.

Diversos autores como Jardim, Semaan e Penna (2016) ,Barbosa (2012), Nguyen et al. (2010) e Carvalho (2011) que trabalharam com o programação de horários baseado em currículo, tiveram como base em suas pesquisas, as restrições definidas por Gaspero, McCollum e Schaerf (2007) na International Timetabling Competition - 2007. As restrições fortes são:

- **Restrição de aulas:** aulas de uma disciplina devem ser agendadas em horários distintos;
- **Restrição de conflitos:** aulas de disciplinas de um mesmo currículo não devem ser agendadas em um mesmo horário;
- **Restrição de ocupação das salas:** duas ou mais aulas distintas não devem ocupar a mesma sala no mesmo horário;
- **Disponibilidade do professor:** se o professor que leciona determinada disciplina não tiver disponibilidade em determinado horário, nenhuma aula da disciplina deve ser alocada para o professor. Além disso, um professor não deve ser alocado em duas ou mais turmas diferentes no mesmo horário.

As restrições fracas são:

- **Restrição de capacidade de sala:** a sala alocada para uma turma deve possuir capacidade igual ou maior exigida pela disciplina ou turma;
- **Restrição de número mínimo de dias:** as aulas de uma disciplina devem estar agendadas em um número mínimo de dias;
- **Restrição de compacidade/Janela de horários:** aulas de disciplinas de um mesmo currículo/semestre devem ser agendadas de maneira adjacente, sem ocorrer janelas entre horários;
- **Restrição de estabilidade de sala:** todas as aulas de uma disciplina e de um currículo devem ocorrer sempre na mesma sala.

O estudo de Willemen (2002), também abordado por Bonutti et al. (2012) define duas outras restrições peculiares ao seu problema, mas que são comuns em qualquer universidade. A primeira restrição é relacionada a alocação de turmas em espaços reservados. Geralmente são turmas que necessitam de laboratórios, salas com equipamentos médicos, equipamentos

esportivos para exercerem atividades práticas. A segunda aborda uma restrição de tempo e espaço. Porque universidades possuem muitos prédios de ensino dentro dos campus. Os professores e estudantes precisam sair de uma aula e ir para outra em um certo espaço de tempo. As duas restrições são mostradas a seguir:

- **Restrição de adequação de salas:** turmas só podem ser lecionadas em salas apropriadas, ou seja, trabalha-se com reserva de salas;
- **Restrição de deslocamentos:** turmas devem estar alocadas minimizando tempo e espaço de deslocamento dos professores e estudantes.

Bonutti et al. (2012) e Barbosa (2012) definem uma tabela comparativa das restrições do CBCT propostas ao longo dos anos. A sigla UD significa Udine, pois foram propostas no PAATAT na Universidade de Udine, Itália. UD1 foi o modelo proposto pelo primeiro evento do ITC 2002. UD2 foi o modelo proposto por Gaspero, McCollum e Schaerf (2007) e é um modelo muito utilizado na resolução do problema. As outras três foram definidas por Bonutti et al. (2012). Ele adiciona uma nova restrição:

- **Restrição de agrupamento de aulas**: aulas que acontecem no mesmo dia devem ser adjacentes sem janelas de horários.

A tabela de comparação dos modelos é mostrada a seguir:

Restrições	**Formulações**				
	UD1	**UD2**	**UD3**	**UD4**	**UD5**
Aulas	F	F	F	F	F
Conflitos	F	F	F	F	F
Ocupação das Salas	F	F	F	F	F
Disponibilidade do Professor	F	F	F	F	F
Capacidade da Sala	1	1	1	1	1
Restrição Mínima de Dias	5	5	-	1	5
Estabilidade da Sala	-	1	-	-	-
Janela de horários	-	2	4	1	2
Deslocamentos	-	-	-	-	2
Adequação das salas	-	-	3	F	-
Agrupamento de Aulas	-	-	-	1	-

Tabela 1 – Comparação entre os principais formulações de restrições definidos para o CBCT

A letra F significa que os autores consideraram como fortes as restrições. Enquanto que os números de 1 a 5 representam restrições fracas, porém com nível de importância, sendo o nível 5 o mais importante. As restrições marcadas com o símbolo - não foram utilizadas nos modelos.

Existem diferentes finalidades em fazer uma função objetivo. É possível, por exemplo, maximizar a preferência de escolha dos horários dos professores Xavier et al. (2013) ou minimizar janelas em horários das turmas. Certas pesquisas como Jardim, Semaan e Penna (2016), Nguyen et al. (2010) e Carvalho (2011) definem a função objetivo para minimizar as violações das restrições fracas pois elas garantem o aperfeiçoamento das soluções: quanto menor for o número de violações dessas restrições, melhor a solução. Para cada restrição, um peso w é aplicado de acordo com a importância daquela restrição. A função objetivo é definida de forma básica como:

$$\text{minimizar} \sum_{i=1}^{n} w_i d_i$$

no qual,

w_i é um peso da i-ésima restrição;

d_i número da violação da i-ésima restrição.

2.4 IBM iLOG CPLEX Studio e a Optimization Programming Language

Optimization Programming Language - OPL é uma linguagem de modelagem criada pela IBM para facilitar a transcrição dos modelos matemáticos em forma de linguagem de programação. A linguagem foi construída com base nas linguagens C++ e Java e contém os mesmos elementos que essas linguagens como estruturas de dados como *arrays*, tuplas e *sets*, funções e procedimentos sobre essas estruturas para facilitar a representação de conjuntos matemáticos em algoritmos. Além disso, fornece ao desenvolvedor facilidade em separar o modelo matemático dos dados de instância.

O manual da IBM sobre o software *IBM iLOG CPLEX Studio* Corporation (2014) mostra como um modelo de programação linear pode ser construído com os seguintes elementos e seguindo as seguintes fases:

- Declaração do modelo: É necessário definir como o modelo será formado. Com OPL, é possível representar os problemas de otimização modelados em programação linear e não linear, utilizando algoritmos de programação com restrições, *simplex* ou *branch and bound* por exemplo. O comando

  ```
  using CP;
  ```

  determina que a ferramenta irá executar um modelo de programação com restrições. Já com o comando

  ```
  using Cplex,
  ```

a ferramenta irá executar um modelo de programação linear inteira ou mista.

- Definição dos conjuntos de dados: a linguagem possui os principais dados primitivos que qualquer linguagem possui: *int*, *float*, *boolean* e *string*. A linguatem também possui as principais estruturas de dados como *array*, *sets* e tuplas. Uma novidade é o tipo de dado *range* que é bem útil para iterar sobre essas estruturas de dados:

- Scripts de pré-processamento: é possível utilizar códigos em Javascript para processar os conjuntos de dados de entrada antes da execução;

- Declaração de variáveis de decisão: as variáveis de decisão podem ser variáveis discretas ou contínuas. A palavra chave utilizada para declarar essa variável é dvar.

- Configurações de busca: um bloco de execução que define as propriedades como tipo de busca: busca em profundidade, busca em largura e limite máximo de tempo de busca.

- Construção da função objetivo e das restrições:

  é possível escrever a expressão matemática a seguir da seguinte forma em OPL:

$$\sum_{i=1}^{n} w_i x_i \tag{2.6}$$

  como:

```
sum {i in 1..n} w[i] * x[i] ;
```

- Scripts de pós-processamento: São manipulações da solução já gerada. Geralmente são procedimentos de exibição dos dados também feitos em Javascript.

2.5 O algoritmo branch and bound

O algoritmo *"branch and bound"* [1] tem como estratégia enumerar soluções em uma estrutura de dados de árvore. O algoritmo explora *branches* da árvore, e cada nó pertencente a um *branch* corresponde a um subproblema, no caso dos problemas de alocação de horários, uma solução parcial da grade de horário. O algorítmo parte do nó raiz , no qual as aulas estão agendadas em horários e salas de valor zero e cria novos *branches* a partir desse nó. Cada nó é responsável por decidir qual horário e sala que a aula que está sendo avaliada naquele *branch* deve assumir. Além disso, as restrições devem validar ou invalidar a decisão desse *branch*.

A função do *bound* no algorítmo é definir limites no qual o *branch* deve pertencer, ou ser descartado caso contrário. O algorítmo possui dois *bounds*: um *lower bound* que calcula um limite inferior para o *branch* e que define que o *branch* aceita somente nós com soluções

[1] tradução: ramificar e limitar

iguais ou superiores ao limite. E o *upper bound* define que a determinada solução obtida é a solução ótima e que nenhuma solução melhor que aquela será produzida. E assim, com esse valor determinado, o algoritmo para de criar *branches* a partir daquele nó analisado, visto que novas *branches* são desnecessárias.

2.6 A metaheurística Iterated Local Search

O ILS é baseado em melhorar uma solução inicial s_0 através de buscas por soluções vizinhas. A vizinhança de soluções $N(S)$ é composta por soluções que contém características semelhantes sejam elas soluções melhores ou piores. A primeira solução s_0 pode ser gerada de forma aleatória ou utilizando outro algoritmo. Em sequência, uma busca local é realizada sobre essa solução inicial, partindo para um outro caminhamento. Isso possibilita o encontro de novas soluções $s*$ e, em caso de melhorias em relação a s_0, esta torna-se a melhor solução encontrada até o momento.

Iterações são realizadas com o objetivo de procurar novas soluções, até que um critério de aceitação seja atendido, como por exemplo o número de iterações máximas. Para cada iteração, métodos de perturbação são executados, ou seja, modificações e movimentações de alocações de recursos em atividades que geram soluções intermediárias s'. Uma nova busca local é realizada em outro ponto inicial sob s', e caso encontre uma solução melhor que $s*$, então gera-se $s*'$ como nova solução ótima local (JARDIM; SEMAAN; PENNA, 2016). Um histórico de conflitos, caso necessário, pode ser construído nessas iterações. Um pseudocódigo foi definido por Lourenco, Olivier e Stutzle (2003) da seguinte forma:

Algoritmo 1 Pseudocódigo ILS

```
s0 ← gerarSolucaoInicial()
s* ← buscaLocal()
repita
    s' ← perturbacao()
    s*' ← buscaLocal()
    s* ← criterioAceitacao(s', s*', historico)
até que critérios de aceitação forem atendidos
```

2.6.1 Gerando a solução inicial

A solução inicial pode ser obtida através de meta-heurísticas gulosas como GRASP ou métodos aleatórios. Para Lourenco, Olivier e Stutzle (2003), utilizar meta-heurística gulosa tem suas vantagens pois, em média, as execuções necessitam de menos processamento computacional. Além disso, oferecem boas soluções iniciais $s*$ possibilitando um bom começo nas buscas de soluções vizinhas. Utilizar algoritmos randômicos proporcionam uma geração rápida de uma solução inicial. Entretanto, acabam utilizando uma maior quantidade de iterações para definir a solução ótima.

2.6.2 O método de busca local

A busca local parte de uma solução inicial s_0, e com movimentações de elementos gera-se novas soluções na vizinhança e tais soluções podem ser soluções válidas ou inválidas. A busca tem como objetivo obter uma solução ótima local minimizada em problemas de minimização ou maximizada para problemas de maximização.

2.6.3 O método de perturbação

As perturbações são inserções, remoções ou alterações de dados já alocados de uma solução. Movimenta-se as atividades alocadas em recursos para novos recursos. No caso do problema CBCT, as movimentações são relacionadas às alocações de disciplinas para professores, horários e salas.

O método de perturbação possui uma propriedade chamada de força. Essa força determina a quantidade de alterações que uma solução deve sofrer, e como consequência aumenta ou diminui o tempo de processamento computacional do procedimento. A cada iteração esse número sofre uma mudança relativa a iteração anterior. Esse procedimento não deve durar muito tempo e nem ser curto. Em caso de um procedimento muito curto, a busca local pode explorar as mesmas soluções, sem explorar soluções diferentes. Em contrapartida, se o procedimento demorar, pode acontecer uma reinicialização da busca , podendo causar um loop (LOURENCO; OLIVIER; STUTZLE, 2003).

2.6.4 Os critérios de aceitação

O critério de aceitação determina quando a solução $s*'$ modificada pela perturbação é aceita ou não. Ele controla o balanço entre intensificação e diversificação de buscas na vizinhança. Intensificação significa o quão profundo deve-se percorrer um caminho e alcançar soluções válidas. Enquanto que diversificação significa a quantidade de caminhos precorridos para buscar soluções válidas.

O trabalho de Lourenco, Olivier e Stutzle (2003) define dois métodos de aceitação. Esses dois métodos são exemplos de métodos extremas de aceitação com intensificação e de aceitação com diversificação. Qualquer método intermediário entre estes podem ser feitos. O método que prioriza a intensificação de buscas permite que a solução seja validada se uma melhor solução for encontrada. O procedimento se chama Better e é definido para um problema de minimização como:

```
Better(s*,s*',history) =  s*' if f(s*') < f(s*) ,  s* otherwize
```

Já este outro é por caminhamento randômico, permitindo que toda solução encontrada seja aceita. Esse procedimento é o método de extrema diversificação. A função é nomeada de RW, *random walk*[2], e é apresentada a seguir:

```
RW(s*,s*',history) = s*'
```

2.7 Trabalhos Relacionados

Os problemas de programação de horários são objetos frequentes de estudos e muitos pesquisadores contribuíram com abordagens diferentes. Entre elas, a proposta de caso de estudos apresentada por Jardim, Semaan e Penna (2016) na Universidade Federal Fluminense com o objetivo de implementar o algoritmo de Iterated Local Search - ILS[3] para resolver o CBCT. Para obter uma solução ótima, os autores definiram a função objetivo para minimizar as penalidades ocorridas por não atender às restrições fracas. Os autores utilizaram uma pequena base de dados de um departamento da universidade para executar o algoritmo. Os tempos de processamento foram calculados e comparados com a ferramenta FET - *Free Timetabling Software* que a própria universidade utilizou para o processamento dos horários daquele período.

Na Universidade de Hacettepe na Turquia, Aladağ e Hocaoğlu (2007) utilizaram algoritmo de busca tabu para resolver o PPHU e construir um software de alocação de professores em turmas. Utilizaram como função objetivo a minimização das penalidades sobre restrições fracas e a solução inicial desse algoritmo é feita utilizando algoritmos gulosos. Também com busca tabu, Nguyen et al. (2010) resolvem o problema do CBCT utilizando várias formas de movimentações soluções vizinhas; *Simple moves, swap moves, Block-changing moves* [4], comparando o desempenho do algoritmo com essas variações.

O algoritmo genético tem como objetivo buscar uma solução ótima até exceder um número de gerações de populações de soluções. Essas populações sofrem alterações através de métodos de *crossovers* e mutações e são substituídas pelas soluções melhoradas. Sutar e Bichkar (2016) utilizaram algoritmos genéticos para resolver o PPHU na Universidade Tecnológica Dr Babasaheb Ambedkar localizada na Índia. Os autores investigaram a influência do tamanho das populações no desempenho e qualidade das soluções. Enquanto que Burke e Petrovic (2002) estudou as vantagens de utilizar uma forma híbrida para solução do PPHEU da Universidade de Nottingham, Inglaterra. Vieira e Macedo (2011) utilizou algoritmos genéticos e função objetivo de minimização de penalidades para construção de uma aplicação que resolvesse o problema de PPHE do Departamento de Computação da Universidade Federal de Sergipe.

Fonseca et al. (2014) utiliza uma metaheurística híbrida, ou seja, dois algoritmos de

[2] caminhamento randômico
[3] Tradução: Busca local Iterada
[4] Tradução:Movimentação simples, movimentação de trocas, movimentos de mudanças em bloco

metauheuristica: o Simulated Annealing [5] junto com ILS para resolver o problema de PPHE. Ambos são procedimentos executados após a geração de uma solução inicial randômica. Phuc, Khang e Nuong (2011) utiliza algoritmos genéticos com *bee algorithm* [6] para resolver o CBCT. Os algoritmos genéticos servem para gerar populações enquanto que o *bee algorithm* efetua uma busca de soluções na vizinhança combinado com busca randômica. Barbosa (2012) compara diferentes combinações de implementação do ILS com meta-heurísticas para a solução do CBCT.

É possível obter melhorias tanto a nível de software quanto a nível de hardware para o problema de PPHE. Yousef et al. (2017) implementa algoritmos genéticos para solucionar PPHU utilizando o framework CUDA para GPU pois este hardware é capaz de processar dados em larga escala. GPU pode explorar grandes tamanhos de populações de soluções e acelerar passos complicados do processo de resolução.

Existe diversos grupos de pesquisas,simpósios, conferências e até mesmo competições sobre o tema. O grupo de estudos *The Scheduling And Timetabling Group* [7] fundado em 2004 na Universidade de Undine, pesquisa soluções usando algoritmos de busca local e abordagem híbrida e produziu publicações como Bellio et al. (2016) e Post et al. (2013). *Practice and Theory of Automated Timetabling (PATAT)* [8], uma comunidade de pesquisadores da área que organiza a Conferência Internacional sobre a Teoria e Prática de Automatização de programação dos horários . A competição *International Timetabling Competition*[9] que com três edições realizadas, sendo a ultima em 2011, encorajou novos pesquisadores para a área, como Lewis, Paechter e McCollum (2007),Gaspero, McCollum e Schaerf (2007) e Post et al. (2013) e são importantes referências para área de pesquisa.

[5] Tradução: Arrefecimento simulado
[6] Tradução: Algoritmo de enxame de abelhas
[7] Tradução: Grupo de pesquisas sobre agendamento e organização de horários
[8] Tradução: Grupo de Teoria e Prática de Automatização de Programação de Horários
[9] Tradução: Competição Internacional de organização de grades de horários

3
Metodologia

Para atingir os objetivos citados neste trabalho, inicialmente foi feito uma revisão bibliográfica estudando o CBCT, comparando diversos modelos de restrições utilizadas. Após isso, foi verificado com detalhes os principais componentes, restrições, variáveis de decisão e função objetivo que envolvem a construção das grades de horários da Universidade Federal de Sergipe. Após a definição desses componentes, foi construído um modelo de programação linear inteira considerando as necessidades especificas para a UFS e uma base de dados com as ofertas de disciplinas da universidade.

O modelo de programação linear foi executado com método exato de *branch and bound* que é incorporado a uma biblioteca do software *IBM iLOG CPLEX Studio*. Por fim, o método heurístico Iterated Local Search foi construído com o objetivo de ser comparado com a abordagem exata. O método escolhido, afirmado por Jardim, Semaan e Penna (2016), é de fácil implementação e eficaz para problemas de programação de horário.

3.1 Formulação do problema

As disciplinas de graduação da Universidade Federal de Sergipe podem ser ofertadas para mais de uma turma, de diferentes currículos. Por exemplo, a disciplina Programação Imperativa possui turmas para Ciência da Computação, Engenharia da Computação e Sistemas de Informação em horários diferentes. Os currículos que agrupam as disciplinas definem que estas não devem conflitar horários. Além disso, os currículos delimitam o início e fim das aulas em um determinado turno. Por exemplo, as aulas ofertadas para Engenharia da Computação devem iniciar e terminar no turno da manhã.

Os professores aptos a lecionar as disciplinas informam suas preferências por determinados horários. Eles podem lecionar dentro dos cinco dias úteis da semana. Cada dia possui oito

horários de duração de duas horas cada, sendo três pela manhã, três pela tarde e dois durante a noite. A seguir, é mostrado uma tabela com a visualização dos horários possíveis a serem alocados para as aulas, baseado em Vieira e Macedo (2011):

Horário / Dia	**SEG**	**TER**	**QUA**	**QUI**	**SEX**
07h-09h	0	8	16	24	32
09h-11h	1	9	17	25	33
11h-13h	2	10	18	26	34
13h-15h	3	11	19	27	35
15h-17h	4	12	20	28	36
17h-19h	5	13	21	29	37
19h-21h	6	14	22	30	38
21h-23h	7	15	23	31	39

Tabela 2 – Relação Dia/Horários letivos da Universidade Federal de Sergipe

As aulas das disciplinas devem ocorrer em salas com capacidade compatível, sem exceder o número. Além disso, algumas disciplinas necessitam de salas especiais como laboratórios e salas de equipamentos médicos. Essas salas são reservadas especialmente para essas disciplinas. Com essas informações, é possível criar um modelo de programação linear inteira. Os elementos de entrada são:

- $i \in A$ elemento do conjunto das aulas ofertadas enumerados de $1, ..., |A|$;
- $p \in P$ elemento do conjunto de professores enumerados de $1, ..., |P|$;
- $c \in C$ elemento do conjunto de currículos enumerados de $1, ..., |C|$;
- $d \in D$ elemento do conjunto de disciplinas enumerados de $1, ..., |D|$;
- $s \in S$ elemento do conjunto de salas enumerados de $1, ..., |S|$;
- $h \in H$ elemento do conjunto de horários letivos enumerados de $0, ..., 39$;
- $mc \in MC, MC \subset C$ os currículos do turno da manhã enumerados de $1, ..., |MC|$;
- $tc \in TC, TC \subset C$ os currículos do turno da tarde enumerados de $1, ..., |TC|$;
- $nc \in NC, NC \subset C$ os currículos do turno da noite enumerados de $1, ..., |NC|$;
- $Q(s)$, informa a capacidade de uma sala s;
- $q(i)$, informa a capacidade de uma aula i;
- $R(s, d)$, informa se a sala s foi reservada para disciplinas d;
- $r(d)$: informa o número de aulas semanais da disciplina d;

- $W_{c,h,p} \in 1..10$, indica o peso que cada professor p tem para lecionar uma aula de currículo c em determinado horário h.

A partir dos dados de entrada, são determinadas seis variáveis, que compõem a formulação matemática:

- $salaPossivel_i$ conjunto de todas as salas no qual a aula i pode ser lecionada. O conjunto é compostos pelas salas que contém a capacidade $Q(s)$ compatível com a capacidade da aula $q(i)$. Compõe também as salas reservadas $R(s,d)$ para certas disciplinas.
- $professorPossivel_{c,d}$ conjunto que identifica o professor possível a lecionar a disciplina d do currículo c.
- $horaPossivel_{p,c,d}$ conjunto de todos os horários que o professor p pode lecionar a disciplina d do currículo c.
- $CH(p)$, função que retorna a soma total da carga horária de um professor p.

As variáveis de decisão são:

- x_i , responsável por alocar uma aula em um horário;
- y_i , responsável por alocar uma aula para um professor;
- z_i , responsável por alocar uma aula em uma sala;

Neste modelo, as restrições são consideradas restrições fortes, exceto a restrição de preferência do professor. A função objetivo do modelo é o acúmulo do produto da preferência do professor em lecionar aula no horário h com a alocação ou não da aula nesse horário:

$$maximizar \sum_{c \in C} \sum_{h \in H} \sum_{p \in P} \sum_{i \in A} W_{c,h,p} * (x_i = h) \tag{3.1}$$

Sujeito a:

Restrição de Carga Horária: Todo professor p deve ter carga horária mínima de 8 horas e máxima de 20 semanais.

$$8 \leq CH(p) \leq 20 \quad \forall p \in P \tag{3.2}$$

Restrição de Preferência do professor: Garantir que o horário alocado seja o desejado pelo professor p. Seja i o índice do conjunto A que é lecionada pelo professor p. Este tem uma preferencia $W_{c,h,p}$ por lecionar a aulas do currículo c, no horário h. Então, a variável de decisão x_i assume o horário h se este pertencer a conjunto $horaPossivel_{p,c,d}$:

$$x_i \in horaPossivel_{p,c,d} \quad \forall i \in A, \forall p \in P, \forall c \in C, \forall d \in D \tag{3.3}$$

Restrição de disponibilidade do professor: Garantir que o professor p tenha somente uma ou nenhuma aula em determinado horário. Seja i e j os índices do conjunto de aulas A e as aulas do mesmo currículo c. Para a comparação dos horários x_i com x_j, assume-se $i \neq j$, evitando uma comparação duplicada dos índices. Assume-se também que o professor p é o professor responsável pela aula em y_i da disciplina d e do currículo c se $p \in professorPossivel_{c,d}$. Então, a quantidade de aulas x_j alocadas no mesmo horário que x_i e lecionada pelo professor y_i deve ser menor ou igual a um:

$$\sum_{j \in A} (x_i = x_j) * (y_i = p) \leq 1 \quad \forall i \in A, \forall p \in P \tag{3.4}$$

Garantir que o professor pode lecionar a disciplina daquele currículo. Seja i o índice do conjunto de aulas A. A aula i deve ser alocada para um professor possível, capaz de lecionar a disciplina d para o currículo c. Então, a variável de decisão y_i assume o professor se este pertencer a conjunto $professorPossivel_{c,d}$:

$$y_i \in professorPossivel_{c,d} \quad \forall i \in A, \forall c \in C, \forall d \in D \tag{3.5}$$

Restrição de ocupação das salas: Garantir que não tenha duas ou mais aulas em uma mesma sala s no mesmo horário. Seja i e j os índices do conjunto de aulas A, podendo ser aulas de currículos iguais ou diferentes. Para a comparação dos horários x_i com x_j, assume-se $i \neq j$, evitando uma comparação duplicada dos índices. Assume-se também que a sala s é a sala responsável pela aula em z_i da disciplina d se $s \in salaPossivel_d$. Então, a quantidade de horários x_j alocadas no mesmo horário que x_i e na sala z_i deve ser menor ou igual a um:

$$\sum_{j \in A} (x_i = x_j) * (z_i = s) \leq 1 \quad \forall i \in A, \forall s \in S \tag{3.6}$$

Restrição de capacidade de salas: Garantir que aula seja alocada em uma sala com capacidade compatível. Seja i o índice do conjunto de aulas A. A aula i deve ser alocada para uma sala possível s com capacidade igual ou superior a capacidade de alunos em uma aula $q(i)$ e pertencente ao conjunto $salaPossivel_d$:

$$z_i \in salaPossivel_i \quad \forall i \in A \tag{3.7}$$

Restrição de estabilidade das salas: Garantir que as aulas de um mesmo currículo c e disciplina d aconteçam sempre na mesma sala s. Seja i e j os índices do conjunto de aulas A, i e j aulas de mesmo currículo e de mesma disciplina. Para a comparação dos horários x_i com x_j, assume-se $i \neq j$, evitando uma comparação duplicada dos índices. Então, z_i deve admitir o mesmo valor de z_j:

$$z_i = z_j \quad \forall i \in A, \forall j \in A \tag{3.8}$$

Restrição de conflitos de aulas: Garantir uma única aula de um currículo em determinado horário. Seja i e j os índices do conjunto de aulas A, aulas do mesmo currículo c. Para a comparação dos horários x_i com x_j, assume-se $i \neq j$, evitando uma comparação duplicada dos índices. Então, a quantidade de aulas x_j alocadas no mesmo horário que x_i do currículo c y_i deve ser menor ou igual a um:

$$\sum_{j \in A} (x_i = x_j) \leq 1 \quad \forall i \in A \tag{3.9}$$

Restrição de turno: Aulas dos currículos do turno da manhã devem ser iniciadas e finalizadas pela manhã. O mesmo é válido para os currículos da tarde e da noite. A restrição definida pelo resto de divisão (representado pelo símbolo %) do horário alocado pela total de horários de um único dia: oito. O resto resultante de 0 , 1 e 2 são os horários da manha, enquanto que os horários da tarde são os restos de números 3 a 5. Já os da noite estão entre 5 a 7.

$$0 \leq (x_i \% 8) \leq 2 \quad \forall i \in A, \forall c \in MC \tag{3.10}$$

$$3 \leq (x_i \% 8) \leq 5 \quad \forall i \in A, \forall c \in TC \tag{3.11}$$

$$5 \leq (x_i \% 8) \leq 7 \quad \forall i \in A, \forall c \in NC \tag{3.12}$$

Restrição mínima de dias: Garantir que as aulas ocorram com a quantidade especificada na oferta. Seja i o índice do conjunto de aulas A. O índice i é uma das repetições definida em $r(d)$. Portanto, a quantidade de aulas alocadas em horários deve ser no mínimo 1 e no máximo o número de repetições daquela disciplina:

$$1 \leq (\sum_{i \in A} x_i) \leq r(d), \forall d \in D \tag{3.13}$$

Restrições de domínio:

$$x_i \in 1, ..., |H|, x_i \in \mathbb{N} \tag{3.14}$$

$$y_i \in 1, ..., |P|, y_i \in \mathbb{N} \tag{3.15}$$

$$z_i \in 1, ..., |S|, z_i \in \mathbb{N} \tag{3.16}$$

A seguir, uma tabela que mostra quais as restrições utilizadas em comparação com os modelos de referência de Gaspero, McCollum e Schaerf (2007) e Bonutti et al. (2012). O símbolo V identifica as restrições utilizadas enquanto que $-$ o contrário:

Restrições	Gaspero 2007	Bonutti 2012	Modelo UFS
Aulas	V	V	V
Conflitos	V	V	V
Ocupação das salas	V	V	V
Disponibilidade do professor	V	V	V
Capacidade da Sala	V	V	V
Restrição Mínima de Dias	V	-	V
Estabilidade de Sala	V	-	V
Compacidade/janela de horários	V	V	-
Deslocamentos	-	V	-
Adequação de Salas	-	V	V
Agrupamento de Aulas	-	V	-
Preferência do Professor	-	-	V
Carga Horária do Professor	-	-	V
Turno das aulas	-	-	V

Tabela 3 – Comparação das restrições usadas no modelo UFS com os modelos bases de Gapero 2007 e Bonutti 2012

3.2 Entrada de dados

Os dados de entrada estão inseridas em um arquivo de texto e os dados são lidos linha a linha. Os dados são tuplas compostas por números e cadeias de caracteres. A seguir um exemplo de entrada de dados da oferta de uma turma de Cálculo I para primeiro período de Engenharia de Materiais com três horários semanais e com capacidade de 50 alunos:

```
<"ENG.MATERIAIS 1", "CÁLCULO I",3,50>
```

Um outro arquivo é composto pelas preferências dos professores em lecionar as turmas em determinados horários. A seguir um exemplo de um professor que tem preferência máxima 10 em lecionar a turma ofertada nos horários 3, 19 e 35 . Caso não seja possível ser alocado nesses horários, é alocado em um horário com menor preferência (nesse exemplo, preferência 9 para os horários 4, 20 e 36). Os valores de preferências dos professores foram inseridos arbitrariamente:

```
<"PROFESSOR 1",  "CÁLCULO I",
"ENG.MATERIAIS 1", 10 , {3,19,35} , 9 , {4,20,36} >
```

Os cenários de testes detalhadas no capítulo seguinte foram definidos supondo a disponibilidade dos prédios da universidade. Cada prédio, ou didática, possui cerca de 30 salas com capacidades diferentes. A seguir um exemplo de entrada de uma sala com capacidade de 50 alunos:

```
<"PREDIO 1 - SALA 01",50>
```

3.3 O algoritmo ILS construído

Seguindo os conceitos do capítulo 2 (ver seção 2.6), foi construído o algoritmo ILS composto pelos métodos de geração inicial randômica, busca local e perturbação. A geração inicial possui características de um algoritmo guloso no qual as escolhas dos horários e salas alocados para as aulas são definitivas. A busca local faz movimentações de aulas para horários de preferência do professor. Três métodos de perturbação foram construídos e dois destes foram baseados em Jardim, Semaan e Penna (2016): um de movimentação simples de aulas para diferentes salas e horários (*singleMove*), movimentação dupla de aulas (*doubleMove*), e troca de horários e salas entre duas aulas (*swapMove*). O critério de aceitação verifica se a solução ótima local resultante de uma busca local é maior que a solução candidata corrente.

3.3.1 Construção da solução inicial

A solução inicial foi gerada randomicamente. Isso significa que um professor, um horário e uma sala foram sorteados a serem atribuídos a uma sala. Caso os elementos sejam validados pelas restrições, então ocorre a alocação da aula. A alocação ocorre em duas fases: na primeira, aloca-se o professor para uma aula, sendo validado pelas restrições de professor lecionar disciplina possível e também a de carga horária mínima. Faz sentido separar em fases pois já é pré determinado durante a construção da base de dados que tal professor lecionará a disciplina específica. A segunda fase fica responsável por identificar professor-horário, aula-horário e aula-sala sendo validada por todas as outras restrições restantes. A seguir, o pseudocódigo referente a construção da solução inicial:

Algoritmo 2 Gerar solução inicial

```
Solução s
para toda aula faça
  aulaAlocada ← false
  repita
    professor ← sortearProfessor()
    se não houver conflitos com professor então
      s.alocarProfessorEmAula(professor, aula)
      aulaAlocada ← true
    fim se
  até que aulaAlocada == true
  aulaAlocada ← false
  repita
    horario ← sortearHorario()
    sala ← sortearSala()
    se não houver conflitos com sala e horário então
      s.alocarAulaEmHorario(horario, aula)
      s.alocarAulaEmSala(sala, aula)
      aulaAlocada ← true
    fim se
  até que aulaAlocada == true
fim para
calcularFunçãoObjetivo(s)
retorne s
```

3.3.2 Método de busca local

A busca local é responsável por partir de uma solução inicial até encontrar uma solução ótima local. Para isso acontecer, movimentações e trocas de horários das aulas devem ocorrer. Logo, o objetivo para o problema é encontrar as aulas que não foram alocadas em horários de preferência do professor e mover essas aulas para um horário de preferência, se este não causar conflitos em restrições. Também é necessário buscar aulas em que o horário foi alocado no horário de menor preferência e tentar mover estas aulas para o horário de maior preferencia. Um histórico de conflitos é construído e preenchido por cada tentativa de movimento sem sucesso. A seguir, o pseudocódigo:

Algoritmo 3 Busca Local

```
Solução s
para toda aula de determinado currículo faça
  se aula não foi alocada em horário de preferencia do professor então
    horario ← selecionarHorarioDeMaiorPreferencia()
    se não houver conflitos com horário então
      s.moverAulaParaHorario(horario, aula)
    se não
      horario ← selecionarHorarioDeMenorPreferencia()
      se não houver conflitos com horário então
        s.moverAulaParaHorario(horario, aula)
      fim se
    fim se
  se não
    horario ← selecionarHorarioDeMaiorPreferencia()
    se não houver conflitos com horario então
      s.moverAulaParaHorario(horario, aula)
    fim se
  fim se
fim para
calcularFunçãoObjetivo(s)
construirHistoricoDeConflitos()
se s melhorou então
  retorne s
fim se
```

3.3.3 Perturbação: SingleMove

Segundo (LOURENCO; OLIVIER; STUTZLE, 2003), *"a perturbação precisa ser suficientemente forte para permitir que a busca local explore diferentes espaço de soluções, mas, também, fraca o suficiente para evitar um reinício aleatório"*. Partindo desse conceito, utiliza-se o histórico para fazer a perturbação com as aulas conflitantes. Isso faz com que a perturbação não modifique todas as aulas já alocadas com sucesso mas também impede que não modifique nenhuma aula, evitando que o algoritmo fique em *loop* infinito em uma vizinhança.

O método de movimentação simples utiliza o histórico para mover as aulas conflitantes para uma sala e um horário qualquer. Isso faz com que a busca local consiga mover a aula para um horário preferencial do professor. Diferentemente da busca local, a perturbação retorna uma solução modificada independentemente se ela é melhor ou pior. O pseudocódigo é mostrado a seguir:

Algoritmo 4 Perturbação: singleMove

```
Solução s
para i ∈ aulas conflitantes faça
    sala ← selecionarSalaRandomicamente()
    horario ← selecionarHorarioRandomicamente()
    se não houver conflitos então
        s.moverAulaParaSala(sala, i)
        s.moverAulaParaHorario(horario, i)
    fim se
    retorne s
fim para
```

3.3.4 Perturbação: DoubleMove

Esse método tem como objetivo mover uma aula com conflito, oriundo do histórico de conflitos, e uma aula sem conflito para horários e salas quaisquer. As aulas devem ser diferentes, porém do mesmo currículo. A seguir, o pseudocódigo que representa esse método:

Algoritmo 5 Perturbação: doubleMove

```
Solução s
garanta i ≠ j
para i ∈ aulas conflitantes faça
    j ← selecionarAulaQualquer
    horario1 ← selecionarHorarioRandomicamente()
    horario2 ← selecionarHorarioRandomicamente()
    sala1 ← selecionarSalaRandomicamente()
    sala2 ← selecionarSalaRandomicamente()
    se não houver conflitos então
        s.moverAulaParaHorario(i, horario1)
        s.moverAulaParaHorario(j, horario2)
        s.moverAulaParaSala(i, sala1)
        s.moverAulaParaSala(j, sala2)
    fim se
fim para
retorne s
```

3.3.5 Perturbação: SwapMove

O método de trocas de posições faz com que a aula com conflito troque horário e sala com uma aula já alocada e sem conflitos. A seguir, o pseudocódigo que representa esse método:

Algoritmo 6 Perturbação: swapMove

Solução s
garanta $i \neq j$
  para $i \in$ aulas conflitantes **faça**
    $j \leftarrow selecionarAulaQualquer$
    se i pode ser alocada no horario de j **e** j pode ser alocado no horario de i **então**
      $s.moverAulaParaHorario(i, horario_j)$
      $s.moverAulaParaHorario(j, horario_i)$
      $s.moverAulaParaSala(i, sala_j)$
      $s.moverAulaParaSala(j, sala_i)$
    fim se
  fim para
  retorne s

3.3.6 Vizinhança de soluções

A partir da solução inicial s0, utiliza-se a busca local para movimentar as aulas para diferentes horários. Cada movimento realizado, gera-se uma nova solução considerada como vizinha. Assim, a vizinhança N(s) é feita tanto com as soluções candidatas válidas e inválidas por causa das restrições. Cada iteração da busca local é um caminhamento diferente. A perturbação serve para transportar a busca para um outro caminhamento com soluções válidas, inválidas ou até o mesmo caminho. A seguir uma representação de uma vizinhança composta por solução inicial s0, soluções inválidas (destacadas em vermelho), soluções candidatas (destacadas em amarelos) e soluções ótimas locais (destacadas em verde):

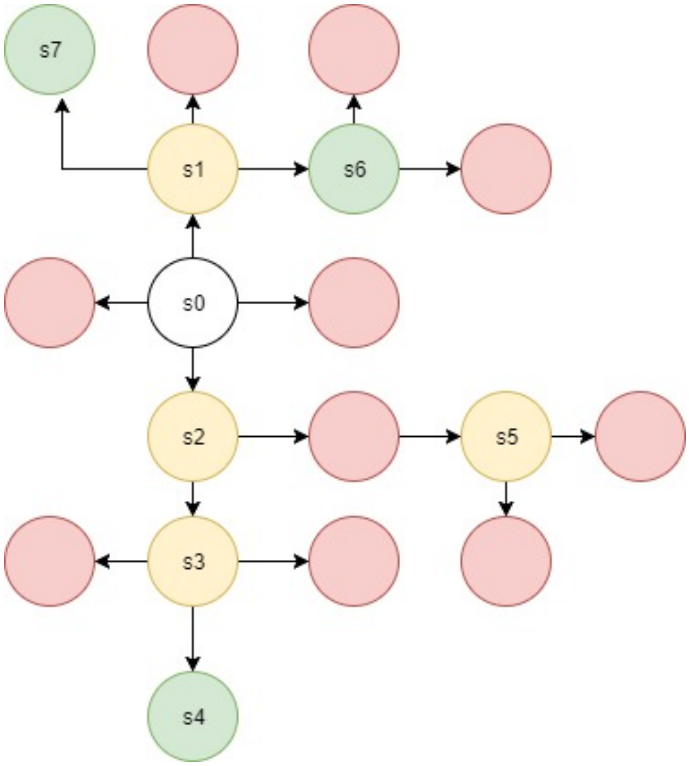

Figura 1 – Vizinhança de soluções

4

Resultados

Foram utilizados dados das ofertas de disciplinas do período letivo 2016.2 da Universidade Federal de Sergipe. Ou seja, 516 aulas de 247 turmas ofertadas, 162 disciplinas diferentes, 158 professores e 54 currículos dos principais cursos matutinos e vespertinos do Centro de Ciências Exatas. Os testes foram feitos em um computador com processador Intel(R) Core(TM) i7, 8GB de *RAM*. O modelo de programação linear inteira foi executado no software IBM iLOG CPLEX Optimization Studio, utilizando o método exato de *branch and bound* para busca de soluções em vizinhanças. Enquanto que o método heurístico utilizado foi o ILS, construído em C++ na ferramenta CodeBlocks.

Três cenários de testes foram definidos, supondo a disponibilidade dos prédios da universidade. Cada prédio, ou didática, possui cerca de 30 salas com capacidades diferentes. Os cenários a serem executados são definidos a seguir:

- Cenário 1 (CN1): alocar 247 turmas em um prédio disponível;
- Cenário 2 (CN2): alocar 247 turmas em dois prédios disponíveis;
- Cenário 3 (CN3): alocar 247 turmas em três prédios disponíveis.

4.1 Execução do método exato

Para cada cenário definido, três execuções com configurações de buscas com tempo diferente : uma, duas e três horas de duração, totalizando nove execuções. A solução de maior valor possível para esta entrada de dados é de 5.160. Na execução do primeiro cenário, não houve solução candidata encontrada. Foram consumidos 9,2Gb de memória. Na execução do segundo cenário, o otimizador conseguiu encontrar uma solução candidata de valor 4.963 como valor

maximizado durante as três horas de execução. Foram consumidos 16Gb de memória durante todo o processo. O gráfico a seguir mostra as soluções encontradas ao longo do tempo:

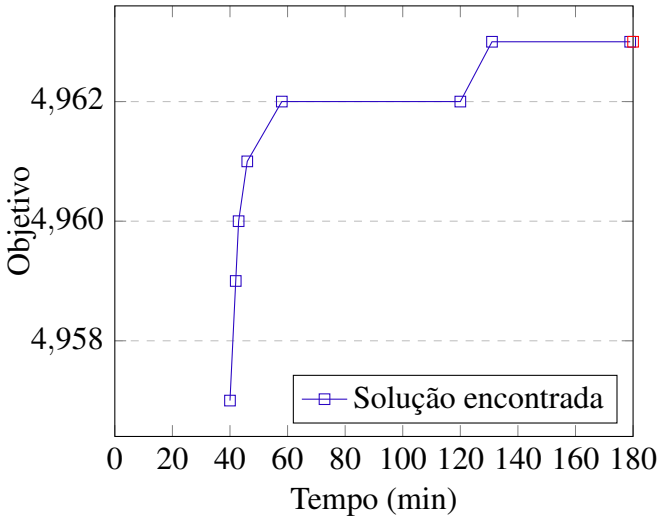

Figura 2 – Soluções encontradas do CN2 em relação ao tempo

E no terceiro cenário, apenas uma solução candidata, de valor 4.957, foi encontrada em três horas de buscas. Foram consumidos 25,6Gb de memória em cada execução. A tabela a seguir mostra em detalhes os resultados das execuções do CN2 e CN3:

Cenário	**Duração de Busca**	**Memória Usada**	**Solução Maximizada**
CN2	1 hora	16GB	4962
CN2	2 horas	16GB	4962
CN2	3 horas	16GB	4963
CN3	1 hora	25,6GB	Não Encontrado
CN3	2 horas	25,6GB	Não Encontrado
CN3	3 horas	25,6GB	4957

Tabela 4 – Resultados das execuções dos cenários do método exato

Nas execuções dos cenários para o método exato, verifica-se a dificuldade da obtenção de solução com um número pequeno de salas no CN1, e que não acaba encontrando solução, pois a quantidade de salas disponíveis não suporta a quantidade de turmas da base de dados. Ao executar o segundo cenário, com uma quantidade maior de salas, obteve-se solução candidata, porém ela não há provas de que é a solução ótima, sendo necessário mais que três horas para ter a comprovação. É visível que o incremento da quantidade de salas proporcionou um aumento significativo no tempo de processamento e recursos utilizados do computador. A execução do terceiro cenário, com uma base de dados superior aos demais cenários, demandou muito mais recursos para encontrar uma solução candidata que também não há provas de que que é a solução ótima.

4.2 Execução do método heurístico

A execução do método heurístico foi realizado com os mesmos cenários e mesma base de dados do método exato, porém com algumas peculiaridades. Cada cenário é executado com os três métodos de perturbação, totalizando nove execuções. E cada execução foi repetida 50 vezes com a finalidade de obter dados estatísticos. Os métodos de perturbações foram executados com um um número fixo de 300 iterações como o critério de parada do ILS, visto que os métodos construídos não encontram melhores soluções ou demoraram a encontrar soluções melhores a partir desse número de iterações.

Durante as execuções dos três métodos de perturbação para o CN1, o ILS não conseguiu completar a alocação das aulas em salas durante a fase de geração de uma solução inicial. O pequeno número de salas e a restrição de estabilidade de salas dificultaram a alocação de aulas de mesma disciplina e currículo nas mesmas salas e portanto o ILS não obteve solução para esse cenário. Já no CN2, obteve-se soluções em todas as execuções. No geral, cada repetição dos métodos *swapMove*, *singleMove* e *doubleMove* duraram 1331 segundos, 1045 segundos e 1176 segundos respectivamente. A seguir os resultados de cada repetição dos métodos de perturbação para o CN2:

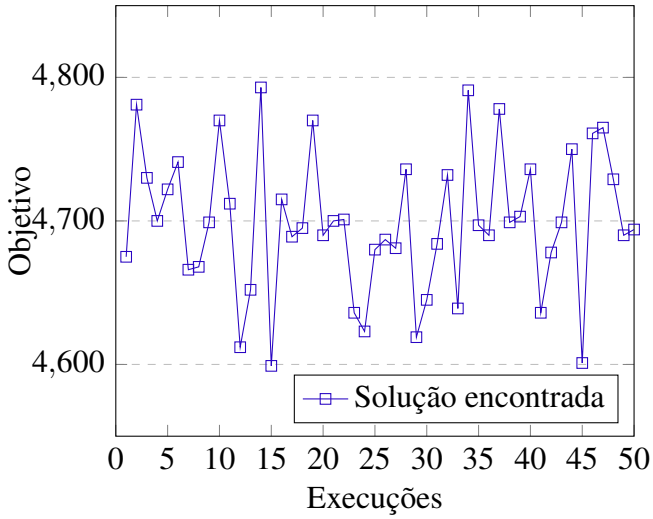

Figura 3 – ILS com swapMove para o cenário 2

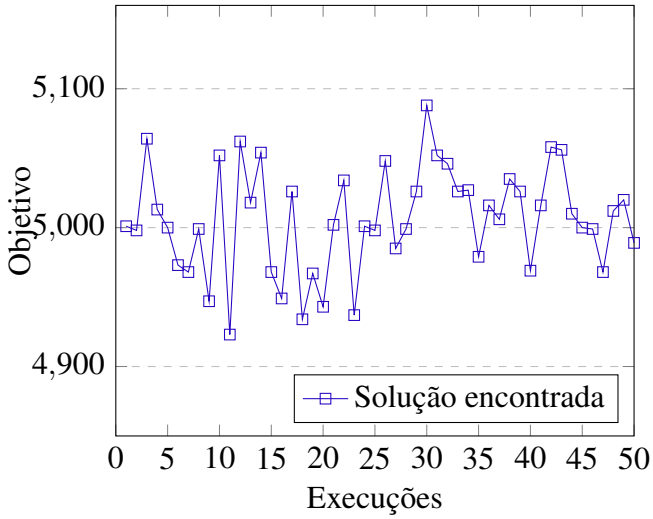

Figura 4 – ILS com singleMove para o cenário 2

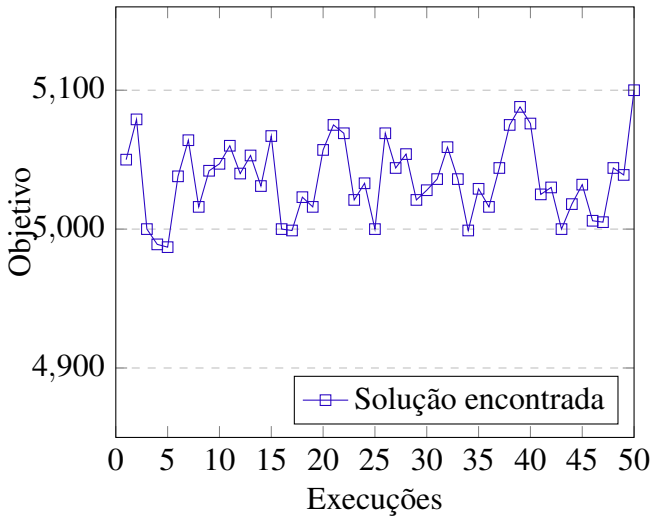

Figura 5 – ILS com doubleMove para o cenário 2

As execuções do CN3 obtiveram soluções com objetivos próximos se comparados ao CN2, porém em tempo reduzido. Cada execução do *swapMove* durou cerca de 720 segundos, enquanto que o *singleMove* durou cerca de 515 segundos e o *doubleMove* durou, no geral, 640 segundos. A tabela a seguir mostra o tempo em que cada repetição durou para cada método e cenário, a média das soluções obtidas nas 50 repetições, desvio padrão e limites de cada execução:

Cenário	Perturbação	Tempo gasto	Objetivo em média	Desvio Padrão	Limite Inferior	Limite Superior
CN2	swapMove	1331 segundos	4698,78	49,5	4599	4793
	singleMove	1045 segundos	5005,74	37,5	4843	5088
	doubleMove	1176 segundos	5036,58	27,12	4987	5100
CN3	swapMove	720 segundos	4700,68	48,42	4606	4791
	singleMove	515 segundos	5003,98	37,28	4895	5070
	doubleMove	640 segundos	5039,92	26,67	4987	5090

Tabela 5 – Comparação dos resultados das execuções dos cenários com diferentes perturbações

O método *doubleMove* mostrou-se mais estável e com menor variação em relação a população de soluções obtidas nas repetições. Em adição, as execuções utilizando esse método obtiveram as soluções com maiores valores. A seguir, um gráfico que mostra a dispersão dos resultados das execuções do CN3 com a média e os limites superiores e inferiores:

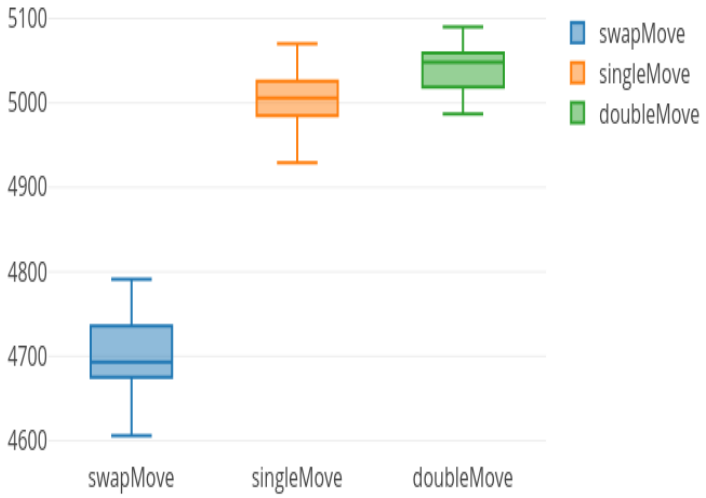

Figura 6 – Resultados do CN3 com os métodos de perturbação

Em comparação com o método exato, no geral, o ILS conseguiu obter soluções em menor tempo. Para o CN1, ambos os métodos não conseguiram obter uma solução válida devido a quantidade insuficiente de salas. No CN2, o método exato conseguiu obter uma solução de valor 4963 como a melhor solução dentro das três horas de execução total. Enquanto que os métodos heurísticos conseguiram em 1176 segundos com a perturbação *swapMove* um valor médio de 4698, variando entre 4599 e 4793. São valores inferiores porém valores considerados próximos ao do método exato. Com os métodos *singleMove* e *doubleMove*, o algorítmo consegue superar os valores do método exato: em média de 5005 e 5039 respectivamente.

O mesmo acontece com o CN3: o método heurístico consegue obter soluções próximas ou melhores que o método exato. A diferença é que ao aumentar a quantidade de salas, o problema se torna mais fácil para o ILS resolver, obtendo soluções em um menor tempo. Já para o método exato, aumentar a quantidade de salas significa também aumentar a quantidade de variáveis do problema.

5

Conclusão

O problema de programação de horários relacionados as universidades é um tema bastante importante na comunidade acadêmica, pois as soluções geralmente são obtidas em tempos computacionais inviáveis e os estudos deste problema ajudam na organização das atividades e recursos fornecidos pelas universidades. A maior contribuição desse trabalho foi a criação do modelo matemático de programação linear específico para a Universidade Federal de Sergipe, que pode servir de base na construção de uma ferramenta automatizada para a construção das grades de horários dos cursos da universidade.

Com os experimentos feitos estima-se que a execução de uma base de dados completa da universidade através do método exato demandaria muito tempo computacional, assim como encontrar uma solução candidata que fosse igual ou próxima a solução ótima do problema. Com isso, a implementação do método heurístico tinha como objetivo executar os mesmos cenários e obter soluções aproximadas das soluções obtidas no método exato. As execuções dos cenários com o método ILS com diferentes métodos de perturbações comprovam que são métodos eficientes, e que utilizaram recursos e tempos computacionais em menores quantidades para a obtenção de soluções próximas ou até melhores das soluções obtidas no método exato.

Durante a realização desse trabalho, houve algumas dificuldades como identificar as restrições e a função objetivo que atendesse as necessidades da UFS. Além disso, entender como o software IBM iLOG CPLEX Studio executa o modelo matemático e também adaptar-se a linguagem de programação OPL. Em contrapartida, o ILS é um método fácil de ser implementado.

Como trabalho futuro, é possível construir uma ferramenta capaz de simular os horários dos professores e/ou organizar os horários das bases curriculares. E também um estudo similar somente com o problema de programação de aulas em salas após a fase de matrículas, pois as aulas são alocadas em salas somente após a confirmação da quantidade de alunos matriculados. Um

outro estudo que pode ser feito é a distribuição das aulas independentes para cada departamento, diminuindo assim a complexidade do problema.

Referências

ALADAğ Çağdaş H.; HOCAOğLU, G. A tabu search algorithm to solve a course timetabling problem. Hacettepe Journal of Mathematics and Statistics, v. 36, n. 1, p. 53 – 64, feb. 2007. Citado na página 25.

BARBOSA, S. H. D. Contribuições para a resolução do problema de programação de cursos universitários baseada em currículos VIA metaheurísticas. Dissertação (Mestrado) — Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, 2012. Citado 6 vezes nas páginas 11, 13, 18, 19, 20 e 26.

BELLIO, R. et al. Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. Computers & Operations Research, v. 65, p. 83–92, 2016. Citado na página 26.

BONUTTI, A. et al. Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. Annals of Operations Research, v. 194, n. 1, p. 59–70, 2012. ISSN 1572-9338. Disponível em: <http://dx.doi.org/10.1007/s10479-010-0707-0>. Citado 3 vezes nas páginas 19, 20 e 31.

BURKE, E. K.; PETROVIC, S. Recent research directions in automated timetabling. European Journal of Operational Research, v. 140, n. 2, p. 266–280, 2002. Disponível em: <http://EconPapers.repec.org/RePEc:eee:ejores:v:140:y:2002:i:2:p:266-280>. Citado na página 25.

CARVALHO, R. de. ABORDAGEM HEURÍSTICA PARA O PROBLEMA DE PROGRAMAÇÃO DE HORÁRIOS DE CURSOS. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, Belo Horizonte, 2011. Citado 2 vezes nas páginas 19 e 21.

COOPER, T. B.; KINGSTON, J. H. The complexity of timetable construction problems. In: ____. Practice and Theory of Automated Timetabling: First International Conference Edinburgh, U.K., August 29–September 1, 1995 Selected Papers. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. p. 281–295. ISBN 978-3-540-70682-3. Disponível em: <http://dx.doi.org/10.1007/3-540-61794-9_66>. Citado na página 18.

CORPORATION, I. ILOG CPLEX Optimization Studio OPL Language Reference Manual. 2014. Citado na página 21.

EVEN, S.; ITAI, A.; SHAMIR, A. On the complexity of time table and multi-commodity flow problems. In: Proceedings of the 16th Annual Symposium on Foundations of Computer Science. Washington, DC, USA: IEEE Computer Society, 1975. (SFCS '75), p. 184–193. Disponível em: <http://dx.doi.org/10.1109/SFCS.1975.21>. Citado 2 vezes nas páginas 11 e 18.

FONSECA, G. H. G. et al. A sa-ils approach for the high school timetabling problem. In: . [S.l.: s.n.], 2014. Citado na página 25.

FREDERICK; HILLIER, G. L. Introduction to Operations Research, 4th Ed. San Francisco, CA, USA: Holden-Day, Inc., 1986. ISBN 0816238715. Citado na página 16.

GASPERO, L. D.; MCCOLLUM, B.; SCHAERF, A. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Tech. rep., Queen's University, Belfast (UK)., 2007. Citado 8 vezes nas páginas 12, 13, 15, 18, 19, 20, 26 e 31.

GOLDBARG, M. C.; LUNA, H. P. L. Otimização Combinatória e Programação Linear - Modelos e Algoritmos. Segunda edição. Rio de Janeiro: Elsevier Editora Ltda, 2005. ISBN 978-85-352-1520-5. Citado 3 vezes nas páginas 13, 16 e 17.

JARDIM, A. M.; SEMAAN, G. S.; PENNA, P. H. V. Uma heurística para o problema de programação de horários: Um estudo de caso. Anais do XLVIII SBPO Simpósio Brasileiro de Pesquisa Operacional, Vitória, p. 12, set. 2016. Citado 7 vezes nas páginas 13, 19, 21, 23, 25, 27 e 33.

LEWIS, R.; PAECHTER, B.; MCCOLLUM, B. A description of the problem model used for track two of the second international timetabling competition. Tech. rep., Cardi University, Wales, UK., 2007. Citado 2 vezes nas páginas 12 e 26.

LOURENCO, H.; OLIVIER, M.; STUTZLE, T. Iterated local search. In: ____. Handbook of Metaheuristics. Boston, MA: Springer US, 2003. p. 320–353. ISBN 978-0-306-48056-0. Disponível em: <http://dx.doi.org/10.1007/0-306-48056-5_11>. Citado 3 vezes nas páginas 23, 24 e 35.

NGUYEN, K. et al. Automating a real-world university timetabling problem with tabu search algorithm. In: 2010 IEEE RIVF International Conference on Computing Communication Technologies, Research, Innovation, and Vision for the Future (RIVF). [S.l.: s.n.], 2010. p. 1–6. Citado 3 vezes nas páginas 19, 21 e 25.

PHUC, N. B.; KHANG, N. T. T. M.; NUONG, T. T. H. A new hybrid ga-bees algorithm for a real-world university timetabling problem. In: 2011 International Conference on Intelligent Computation and Bio-Medical Instrumentation. [S.l.: s.n.], 2011. p. 321–326. Citado na página 26.

POST, G. et al. The third international timetabling competition. Annals of Operations Research, Springer US, v. 239, n. 1, p. 69–75, 2013. ISSN 0254-5330. Citado na página 26.

SANTOS, H. G.; SOUZA, M. J. F. Programação de horários em instituições educacionais: formulações e algoritmos. Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional - SBPO, Fortaleza - CE, p. 2827–2882, aug. 2007. Citado 2 vezes nas páginas 12 e 18.

SUTAR, S. R.; BICHKAR, R. S. An application of genetic algorithm for university course timetabling problem. International Journal of Applied Information Systems, v. 11, n. 3, aug. 2016. ISSN 2249-0868. Citado na página 25.

VIEIRA, F.; MACEDO, H. Sistema de alocação de horários de cursos universitários: Um estudo de caso no departamento de computação da universidade federal de sergipe. SCIENTIA PLENA v. 7, n. 3 (2011), mar. 2011. Citado 2 vezes nas páginas 25 e 28.

WILLEMEN, R. J. School timetable construction : algorithms and complexity. Dissertação (Mestrado) — Technische Universiteit Eindhoven, Eindhoven, 2002. Citado 2 vezes nas páginas 12 e 19.

WINSTON, W. L.; GOLDBERG, J. B. Operations Research. Applications and Algorithms. 4th. ed. [S.l.]: Brooks/Cole, 2004. Hardcover. ISBN 0534423620. Citado na página 17.

WREN, A. Scheduling, timetabling and rostering — a special relationship? In: ____. Practice and Theory of Automated Timetabling: First International Conference Edinburgh, U.K., August 29–September 1, 1995 Selected Papers. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996. p. 46–75. ISBN 978-3-540-70682-3. Disponível em: <http://dx.doi.org/10.1007/3-540-61794-9_51>. Citado na página 11.

XAVIER, B. M. et al. Proposta de alocação de horários de professores e turmas em instituições de ensino superior utilizando uma heurística vns/vnd. Anais do XLVS Simpósio Brasileiro de Pesquisa Operacional - SBPO, Natal - RN, 2013. Citado na página 21.

YOUSEF, A. H. et al. A gpu based genetic algorithm solution for the timetabling problem. IEEE Conference Publications, p. 103 – 109, jan. 2017. Citado na página 26.